# Operating Systems
## (1DT020 & 1TT802)

# Lecture 1
# Introduction

April 3, 2008

Léon Mugwaneza

http://www.it.uu.se/edu/course/homepage/os/vt08

# Who am I?

- **Léon Mugwaneza**
  - **Visiting lecturer**
  - **Senior lecturer**

    **Computer science departmennt**

    **ESIL/université de la Méditerranée**

    **Marseille, France**

# Where is Marseille?

# ESIL, Luminy campus, and Marseille (view from Mt Puget)

# Goals for Today

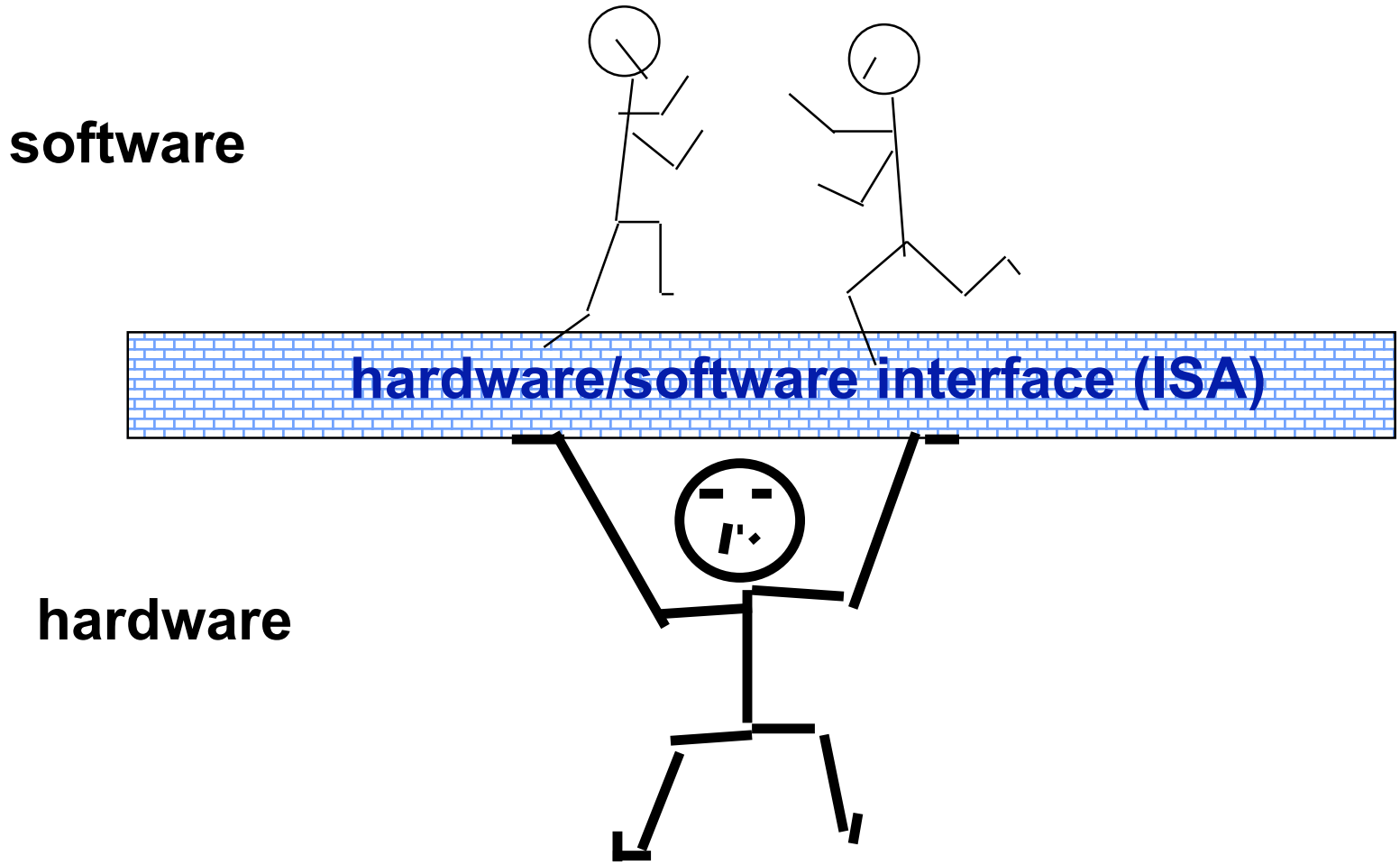
- What is an Operating System?
  - And – what is it not?
- Examples of Operating Systems design
- What is in this course ?
- Why study Operating Systems?
- Also "How does this course operate?"

Note: Some slides and/or pictures in the following are adapted from slides ©2005 Silberschatz, Galvin, and Gagne, others from Kubiatowicz - CS162 ©UCB Fall 2007 (University of California at Berkeley)

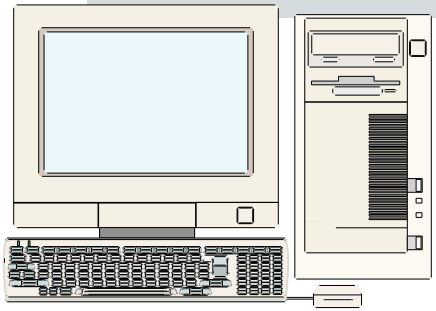
# What is an Operating System ?

- **Software that makes the computer easy to use**
  - **What is software ?**
  - **What is a computer ?**

# A Computer System = Hardware + Software

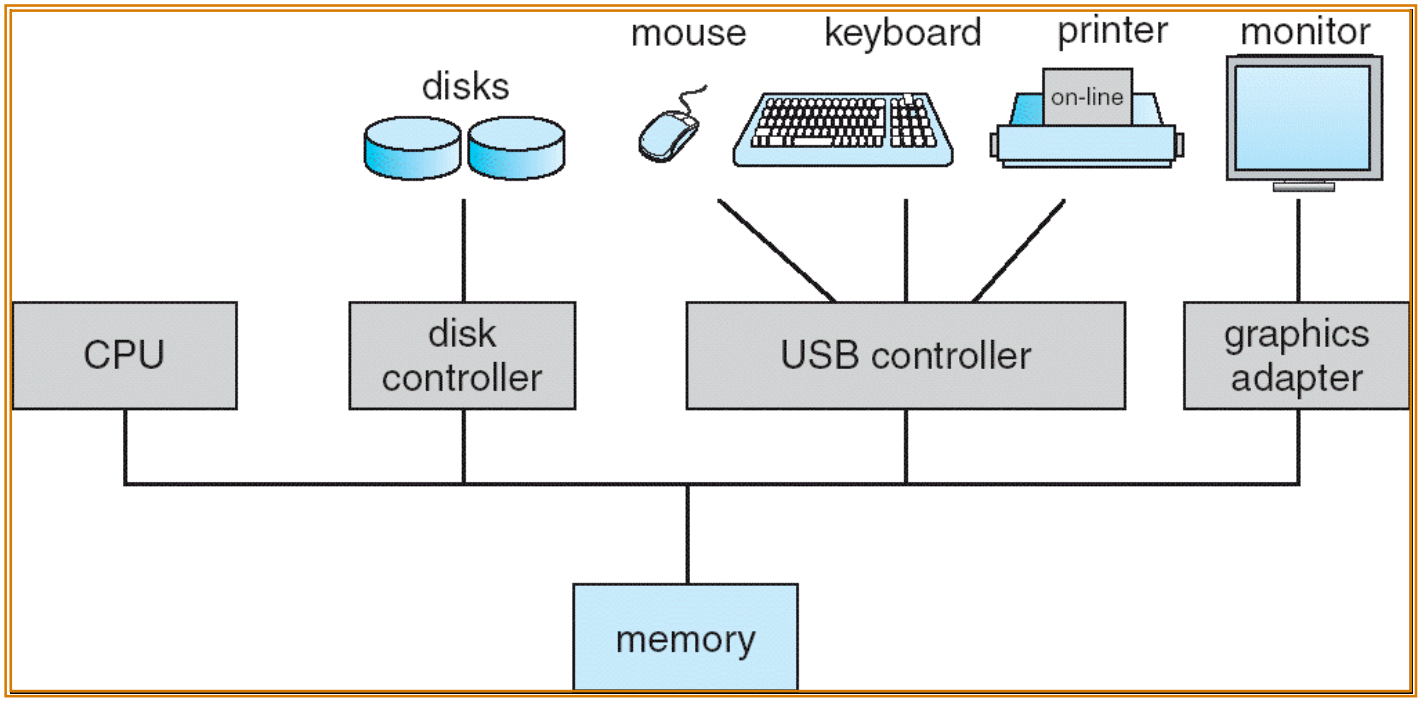


# Computing Devices Everywhere

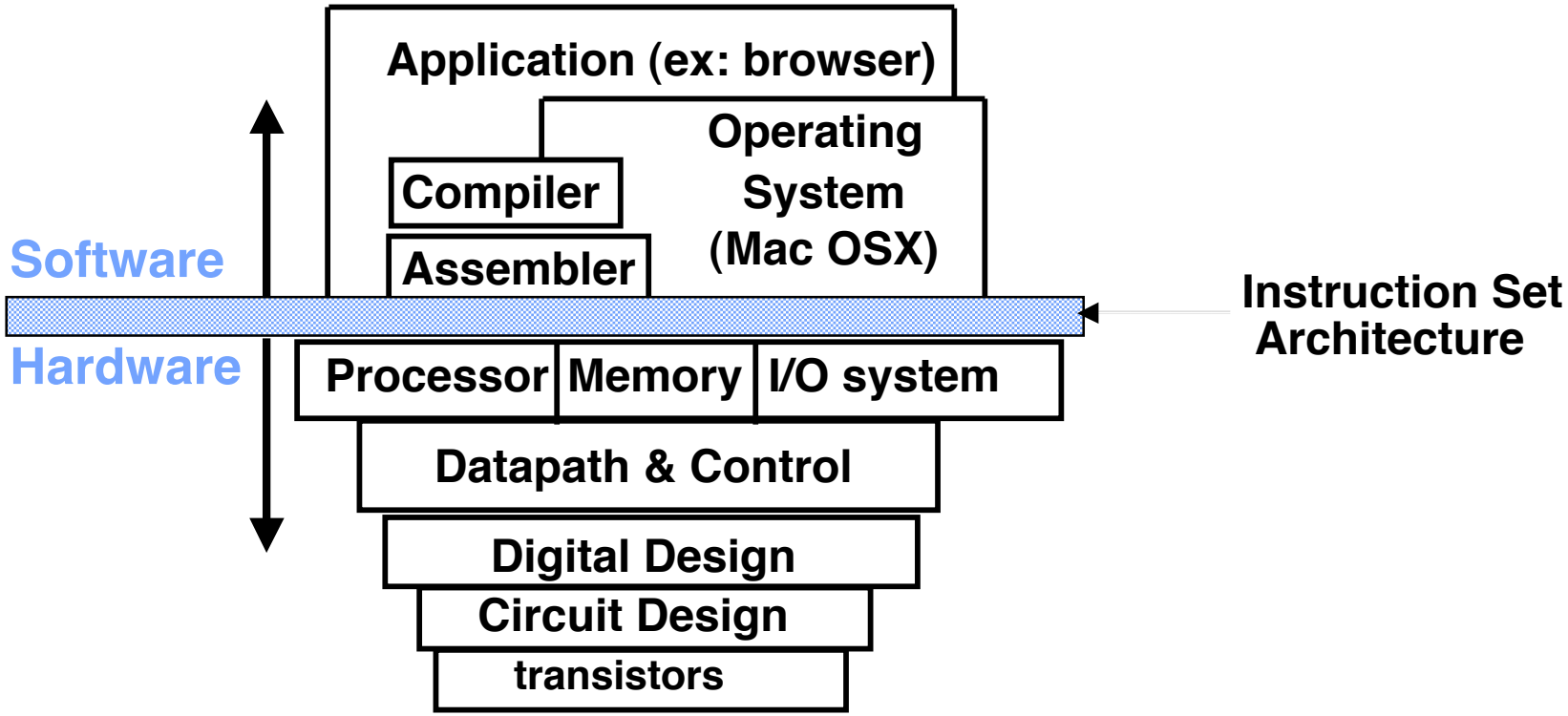


# Computer System hardware Organization

- **Computer-system operation**
  - **One or more CPUs, device controllers connect through common bus providing access to shared memory**
  - **Concurrent execution of CPUs and devices competing for memory cycles**



# General purpose Computer Systems structure

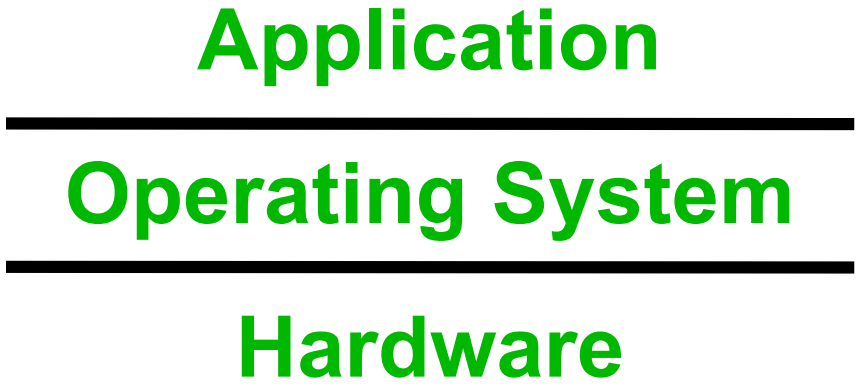



- Several levels of abstraction
- Often, "System software" is used to name sofware between applications and hardware

# How do we tame complexity?

- Every piece of computer hardware is different
  - Different CPU
    - » Pentium, PowerPC, ColdFire, ARM, MIPS
  - Different amounts of memory, disk, …
  - Different types of devices
    - » Mice, Keyboards, Sensors, Cameras, Fingerprint readers
  - Different networking environment
    - » Cable, DSL, Wireless, Firewalls,…
- Questions:
  - Does the programmer need to (re)write his on code for common tasks?
  - Does the programmer need to write a single program that performs many independent activities?
  - Does every program have to be altered for every piece of hardware?
  - Does a computer system run only one user program at time?
  - Does a faulty program crash everything?
  - Does every program have access to all hardware?

# OS Tool: Virtual Machine Abstraction

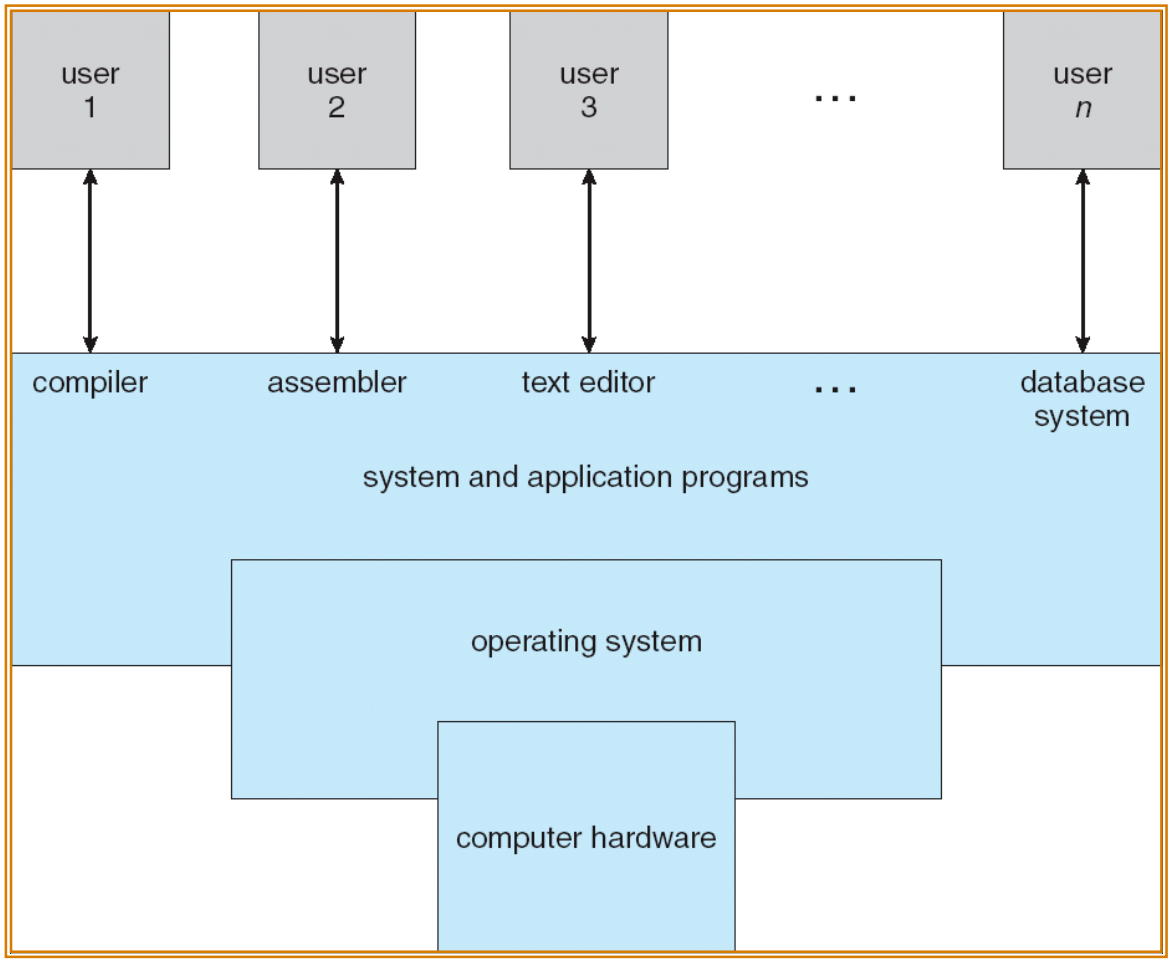



Virtual Machine Interface

Physical Machine Interface

- **Software Engineering Problem:**
  - Turn hardware/software peculiarities into what programmers want/need
  - Optimize for convenience, utilization, security, reliability, etc…
- **For Any OS area (e.g. file systems, virtual memory, networking, scheduling):**
  - What's the hardware interface? (physical reality)
  - What's the application interface? (nicer abstraction)

# Four (3?) Components of a Computer System




**Definition:** ***An operating system implements a virtual machine that is (hopefully) easier and safer to program and use than the raw hardware.***

# What does an Operating System do?

- **Silerschatz and Gavin:**
  **"An OS is Similar to a government"**
  - **Raises the question: does a government do anything useful by itself?**
- **Coordinator and Traffic Cop:**
  - **Manages all resources**
  - **Settles conflicting requests for resources**
  - **Prevent errors and improper use of the computer**
- **Facilitator:**
  - **Provides facilities that everyone needs**
  - **Standard Libraries, Windowing systems**
  - **Make application programming easier, faster, less error-prone**
- **Some features reflect both tasks:**
  - **E.g. File system is needed by everyone (Facilitator)**
  - **But File system must be Protected (Traffic Cop)**

# What is an Operating System,… Really?

- **Most Likely:**
  - **Memory Management**
  - **I/O Management**
  - **CPU Scheduling**
  - **Communications? (Does Email belong in OS?)**
  - **Multitasking/multiprogramming?**
- **What about?**
  - **File System?**
  - **Multimedia Support?**
  - **User Interface?**
  - **Internet Browser? ☺**
- **Is this only interesting to Academics??**

# Course Administration

- **Instructor:**

  **Léon Mugwaneza (lectures and labs)**

  **Office : pol_1340 - email : leon.mugwaneza@it.uu.se**

  **Office hours : to be announced next week**

- **Course homepage : http://www.it.uu.se/edu/course/homepage/os/vt08**

- **Labs : 3 labs**
  - **1st lab on April 18 (material on the course homepage 1 week before)**
  - **Deadlines will be announced later**

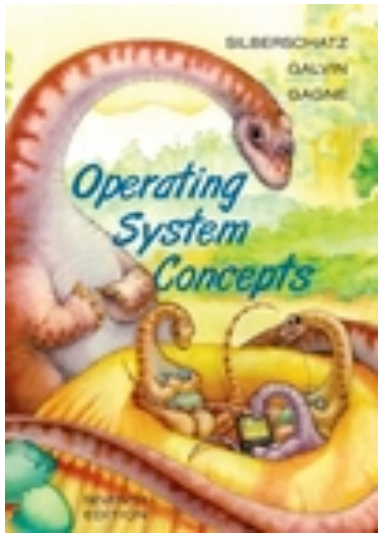
- **Evaluation**
  - **1 written exam - closed books : 4.5 points (for both 1DT020 & 1TT802)**
  - **Labs : 3 points for 1DT020 , 1.5 points for 1TT802**

- **Course Registration :**
  - **Tick your name on one of the 2 lists**

  **or**
  - **Fill in your name (and other data) on the appropriate sheet (1DT020 or 1TT802)**

# Textbook

- **Text: Operating Systems Concepts, 7th Edition Silberschatz, Galvin, Gagne**




- **Online supplements**
  - **See "Student companion site" link on course homepage**
  - **Includes Appendices, sample problems, etc**

# Topic Coverage

**Textbook: Silberschatz, Galvin, and Gagne,**
***Operating Systems Concepts*, 7th Ed., 2005**

- **1 lecture:** **Introduction (what is an operating system ?)**
- **2 lectures:** **Process Control and Threads, scheduling**
- **3 lectures:** **Process communication and Synchronization**
- **2 lectures:** **Virtual memory (mechanism and policies)**
- **3 lectures:** **File systems and mass storage devices**
- **1 lecture:** **I/O systems**
- **1 lecture:** **Protection and security**

# Operating System Definition (Cont.)

- No universally accepted definition
- "Everything a vendor ships when you order an operating system" is good approximation
  - But varies widely
- "The one program running at all times on the computer" is the kernel.
  - Everything else is either a system program (ships with the operating system) or an application program

# What if we didn't have an Operating System?

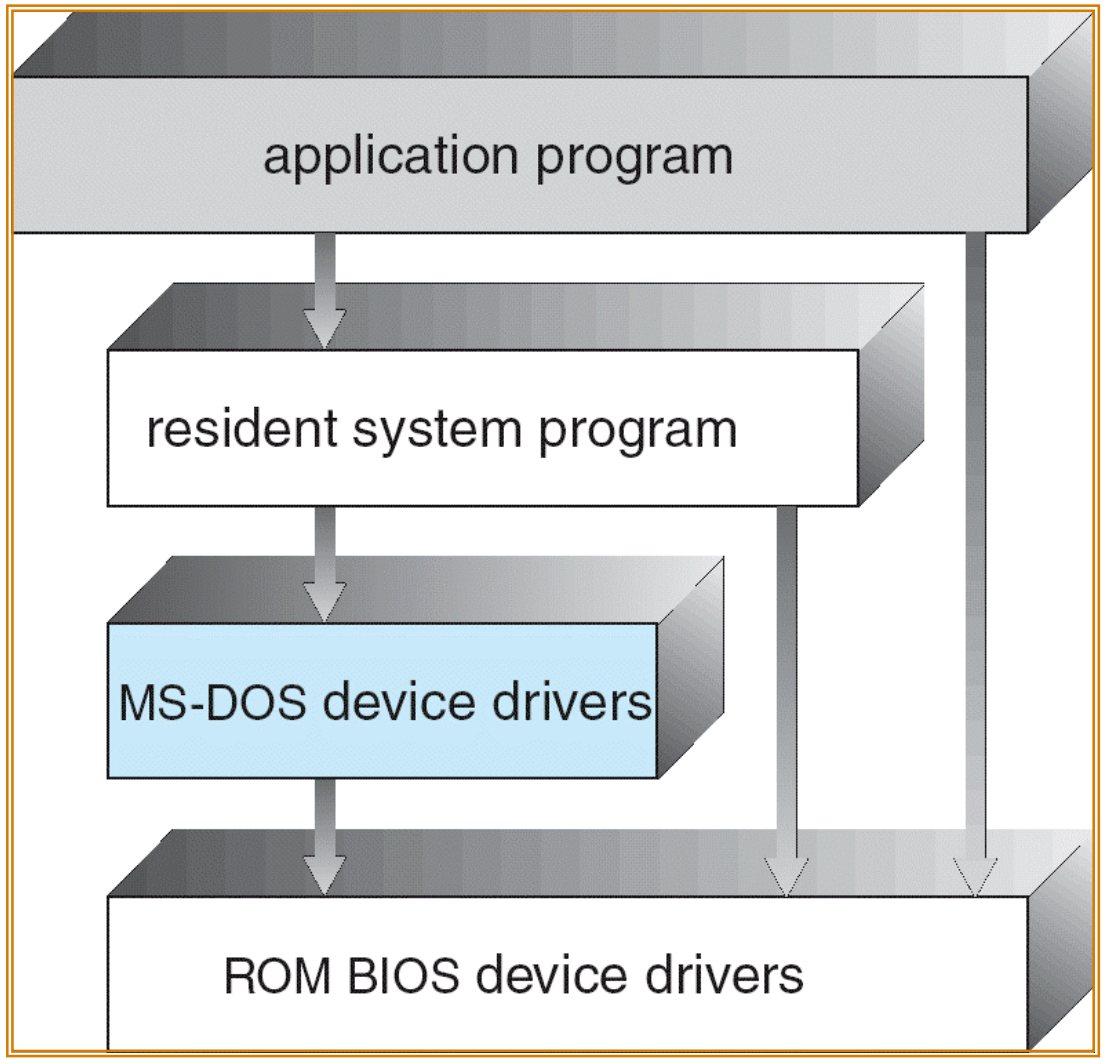
- Source Code⇒Compiler⇒Object Code⇒Hardware
- How do you get object code onto the hardware?
- How do you print out the answer?
- Once upon a time, had to Toggle in program in binary and read out answer from LED's!

Altair 8080

# Simple OS: What if only one application?

- **Examples:**
  - **Very early computers**
  - **Early PCs**
  - **Embedded controllers (elevators, cars, etc)**
- **OS becomes just a library of standard services**
  - **Standard device drivers**
  - **Interrupt handlers**
  - **Math libraries**

# MS-DOS Layer Structure



# More complex OS: Multiple Applications

- **Full Coordination and Protection**
  - **Manage interactions between different users**
  - **Multiple programs running simultaneously**
  - **Multiplex and protect Hardware Resources**
    - » **CPU, Memory, I/O devices like disks, printers, etc**
- **Facilitator**
  - **Still provides Standard libraries, facilities**

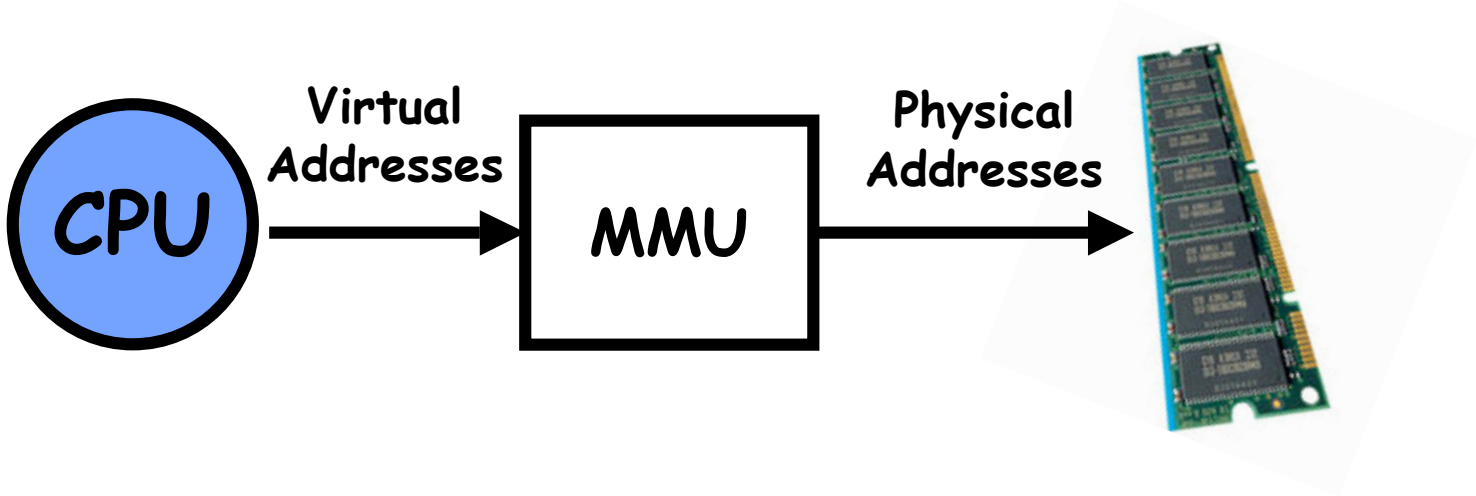
- **Would this complexity make sense if there were only one application that you cared about?**

# Example: Protecting Programs from Each Other

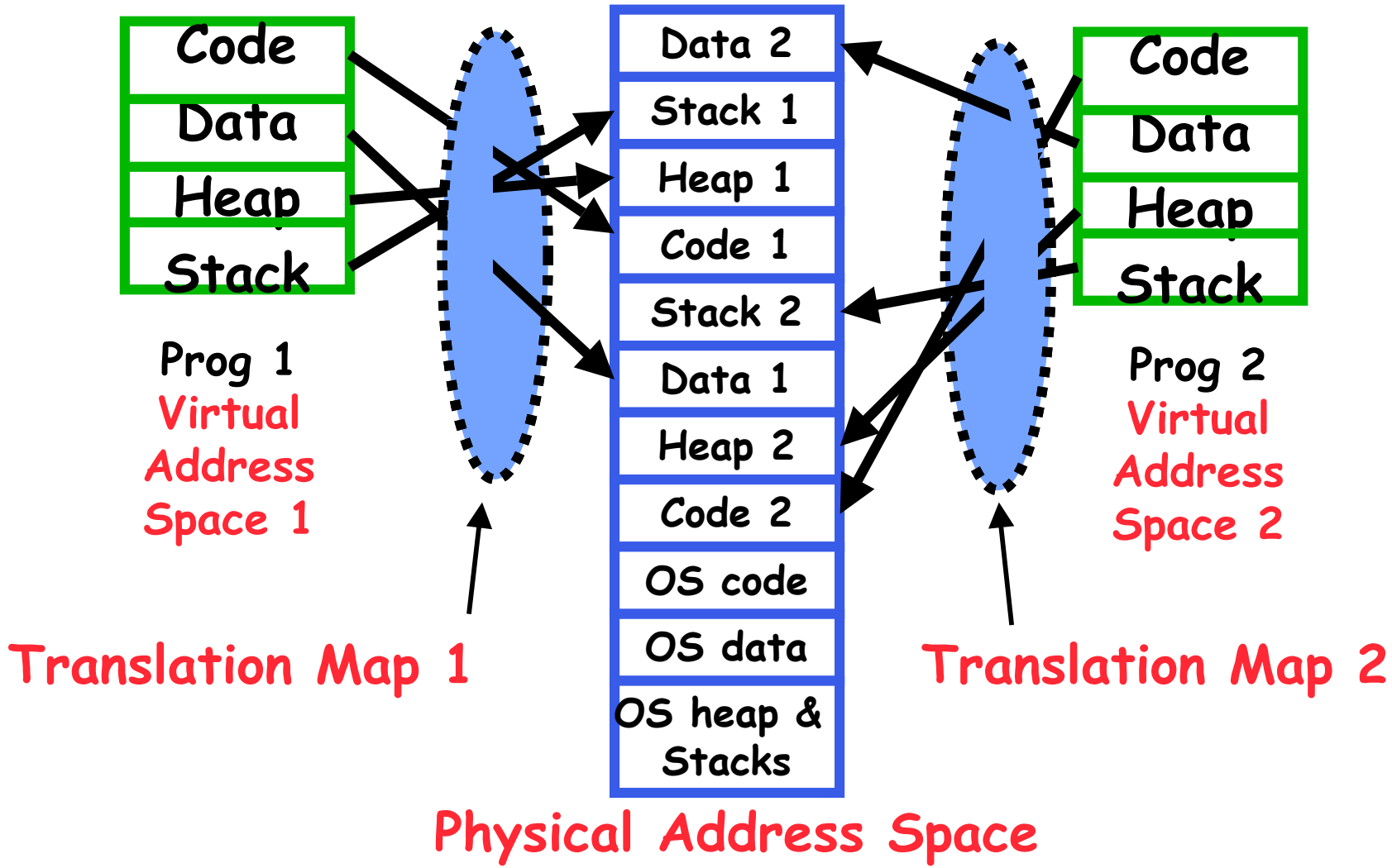
- **Problem: Run multiple applications in such a way that they are protected from one another**
- **Goal:**
  - **Keep User Programs from Crashing OS**
  - **Keep User Programs from Crashing each other**
  - **[Keep Parts of OS from crashing other parts?]**
- **(Some of the required) Mechanisms:**
  - **Address Translation**
  - **Dual Mode Operation**
- **Simple Policy:**
  - **Programs are not allowed to read/write memory of other Programs or of Operating System**

# Address Translation

- **Address Space**
  - A group of memory addresses usable by something
  - Each program  and kernel has potentially different address spaces.
- **Address Translation:**
  - Translate from Virtual Addresses (emitted by CPU) into Physical Addresses (of memory)
  - Mapping *often* performed in Hardware by Memory Management Unit (MMU)



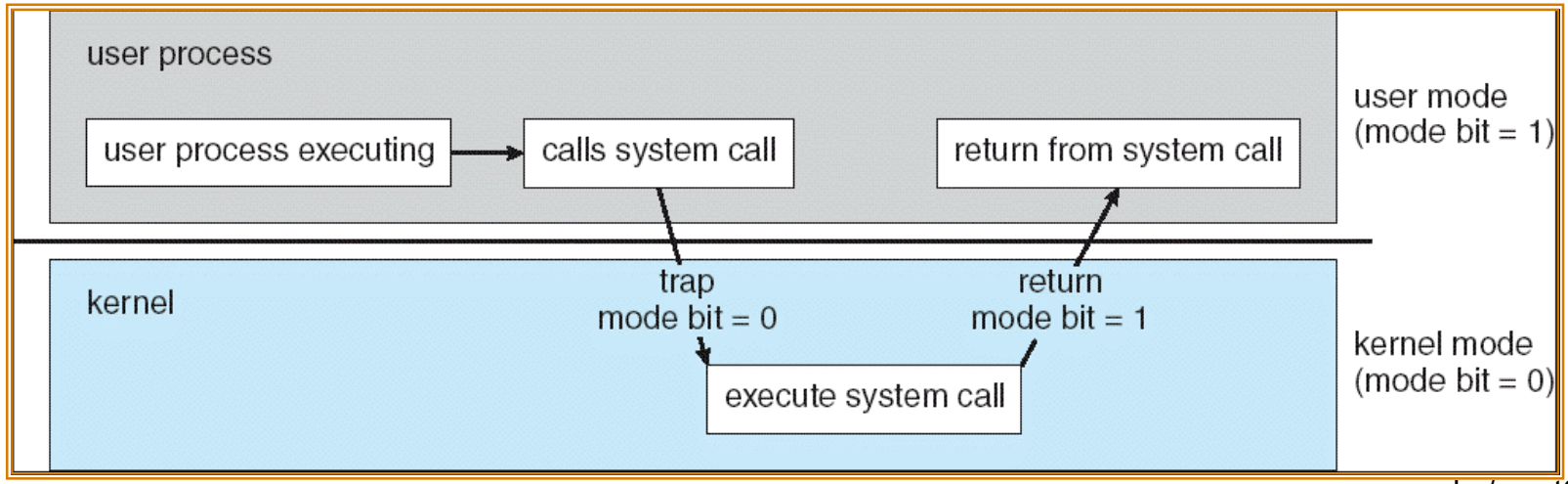
# Example of Address Translation



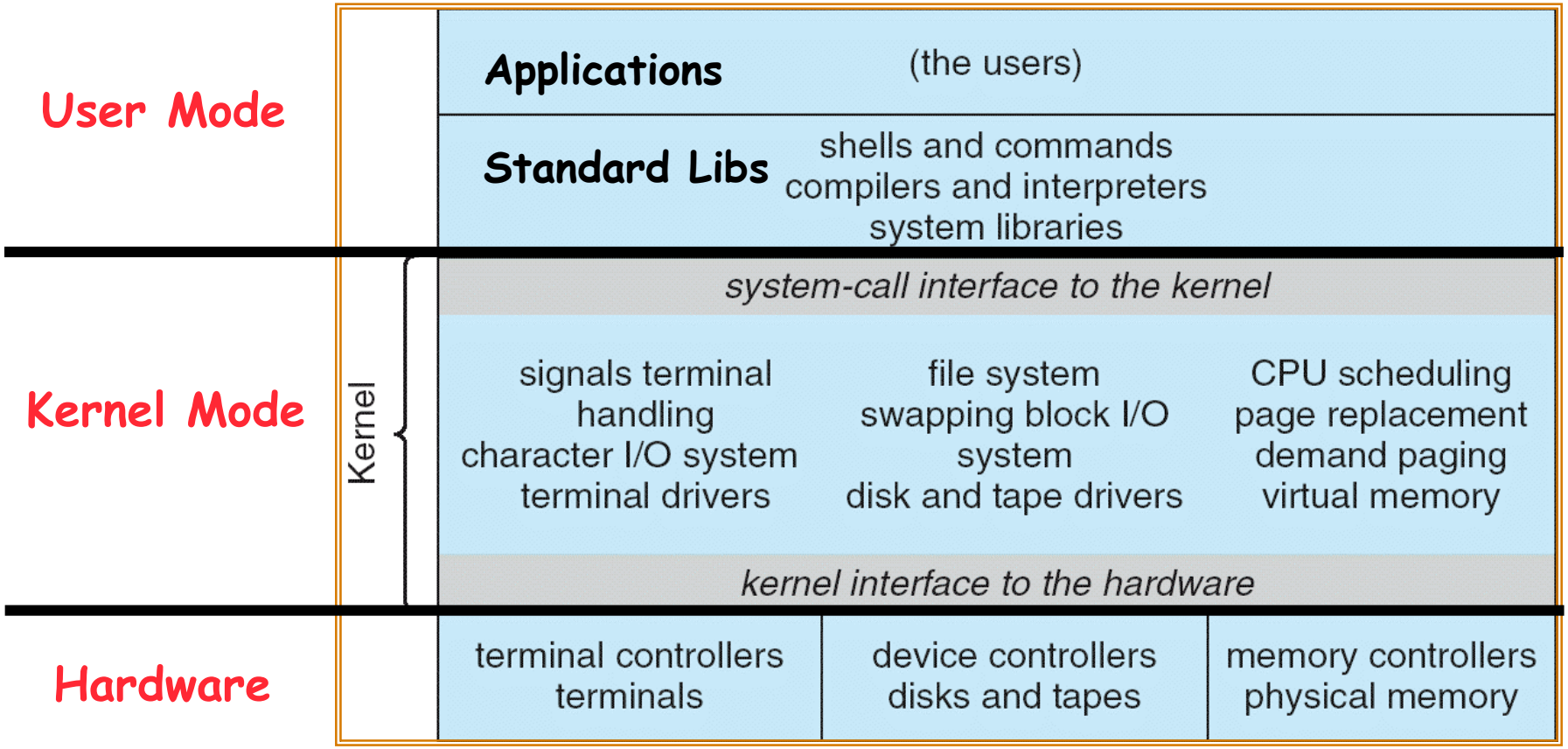

- **Translation helps protection:**
  - Control translations, control access
  - Should Users be able to change Translation map???

# Dual Mode Operation

- **Hardware provides at least two modes:**
  - **"Kernel" mode (or "supervisor" or "protected")**
  - **"User" mode: Normal programs executed**
- **Some instructions/ops prohibited in user mode:**
  - **Example: cannot modify page tables in user mode**
    - » **Attempt to modify ⇒ Exception generated**
- **Transitions from user mode to kernel mode:**
  - **System Calls, Interrupts, Other exceptions**



# UNIX System Structure



# OS Systems Principles

- **OS as illusionist:**
  - Make hardware limitations go away
  - Provide illusion of dedicated machine with infinite memory and infinite processors
- **OS as government:**
  - Protect users from each other
  - Allocate resources efficiently and fairly
- **OS as complex system:**
  - Constant tension between simplicity and functionality or performance
- **OS as history teacher**
  - Learn from past
  - Adapt as hardware tradeoffs change

# Why Study Operating Systems?

- **OS are complex systems:**
  - **How can you manage complexity for future projects?**
- **Buying and using a personal computer:**
  - **Why different PCs with same CPU behave differently**
  - **How to choose a processor (Opteron, Itanium, Celeron, Pentium, Hexium)? [ Ok, made last one up ]**
  - **Should you get Windows XP, Vista, Linux, Mac OS …?**
- **Security, viruses, and worms**
  - **What exposure do you have to worry about?**
- **Discover what is in the black box ! ☺**

# Summary

- Operating systems provide a virtual machine abstraction to handle diverse hardware
- Operating systems coordinate resources and protect users from each other
- Operating systems simplify application development by providing standard services
- Operating systems can provide an array of fault containment, fault tolerance, and fault recovery